| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/804,735 | 03/19/2004 | Li Long | INTEL/18495 | 8282 |

34431          7590          12/28/2007
HANLEY, FLIGHT & ZIMMERMAN, LLC
150 S. WACKER DRIVE
SUITE 2100
CHICAGO, IL 60606

| EXAMINER |
|---|
| WANG, BEN C |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2192 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 12/28/2007 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| | Application No. | Applicant(s) |
|---|---|---|
| **Office Action Summary** | 10/804,735 | LONG ET AL. |
| | Examiner | Art Unit | |
| | Ben C. Wang | 2192 | |

*– The MAILING DATE of this communication appears on the cover sheet with the correspondence address –*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE <u>3</u> MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
  Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on <u>04 October 2007</u>.

2a)☒ This action is **FINAL**.      2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) <u>1-10 and 12-30</u> is/are pending in the application.

     4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) <u>1-10,16-18,20-22,25-27 and 30</u> is/are rejected.

7)☐ Claim(s) <u>12-15,19, 23-24, and 28-29</u> is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

     Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

     Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

     a)☐ All    b)☐ Some * c)☐ None of:

       1.☐ Certified copies of the priority documents have been received.

       2.☐ Certified copies of the priority documents have been received in Application No. _____.

       3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

     * See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
     Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413)
     Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

## DETAILED ACTION

1.     Applicant's amendment dated October 4, 2007, responding to the Office action mailed July 3, 2007 provided in the rejection of claims 1-30, wherein claims 1, 4-5, 17, 19, and 25 are amended, claim 11 is canceled.

Claims 1-10 and 12-30 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims rejection have been fully considered but are moot in view of the new grounds of rejection – see *Haab et al.*, art made of record, as applied hereto.

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a).

Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

### Response to Arguments

2.      Applicant's arguments filed on October 4, 2007 have been fully considered but they are not persuasive.

### In the remarks, Applicant argues that:

a)      The Office action contends, that the components of the apparatus recited in claim 17 "are not physical things" (see Remarks on P. 9, 3$^{rd}$ Par.).

### Examiner's response:

a)      The examiner would suggest applicant(s) include, for example, "a processor that is used to execute software instructions" (as stated in the Remarks on P. 9, 3$^{rd}$ Par., lines 8-10).

### Claim Rejections – 35 USC § 101

35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

3.      Claims 17-24 are rejected under 35 U.S.C 101 because the claims are directed to non-statutory subject matter.

4.    **In claim 17**, an "instruction analysis module", a "cost estimation module",
and a "partition generator", are being cited; however, it appears that the
"instruction analysis module", the "cost estimation module", and the "partition
generator" would reasonably be interpreted by one of ordinary skill in the art as
computer listings per se, are not physical "things". They are neither computer
components nor statutory processes, as they are not "act" being performed. Such
claimed computer programs do not define any structural and functional
interrelationships between the computer program and other claimed elements of
a computer which permit the computer program's functionality to be realized. In
contrast, a claimed computer readable medium encoded with a computer
program is a computer element which defines structural and functional
interrelationships between the computer program and the rest of the computer
which permit the computer program's functionality to be realized, and is thus
statutory. Accordingly, it is important to distinguish claims that define descriptive
material per se from claims that define statutory inventions. (See MPEP
2106.01(I))


5.    **In claim 18**, a "redundant module" and a "mutual exclusion lock module"
are being cited; however, it appears that the "redundant module" and the "mutual
exclusion lock module" would reasonably be interpreted by one of ordinary skill in
the art as computer listings per se, are not physical "things". They are neither
computer components nor statutory processes, as they are not "act" being
performed. Such claimed computer programs do not define any structural and

functional interrelationships between the computer program and other claimed elements of a computer which permit the computer program's functionality to be realized. In contrast, a claimed computer readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of the computer which permit the computer program's functionality to be realized, and is thus statutory. Accordingly, it is important to distinguish claims that define descriptive material per se from claims that define statutory inventions. (See MPEP 2106.01(I))

6.     **As to claims** 21-24, they are merely further recited as the computer program product per se, thus, do not cure the deficiency of base claim 17, and also rejected under 35 U.S.C. 101 as set forth above.

7.     **As to claims** 19-20, they are merely further recited as the computer program product per se, thus, do not cure the deficiency of base claims 17-18, and also rejected under 35 U.S.C. 101 as set forth above.

*Claim Rejections – 35 USC § 103(a)*

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8.      Claims 1-5, 8-9, 16, 17, 22, 25-26, and 30 are rejected under 35 U.S.C.

103(a) as being unpatentable over Tang et al. (*Thread Partitioning and*

*Scheduling Based on Cost Model, 1997, ACM*) (hereinafter 'Tang') in view of

Haab et al. (*Sec. 4.1 Managing Lock Contention, Large And Small Critical*

*Sections, Chap. 4, Synchronization, Developing Multithreaded Applications: A*

*Platform Consistent Approach, Intel® Corporation, March, 2003*) (hereinafter

'Haab' - art made of record)

9.      **As to claim 1** (Currently Amended), Tang discloses a method comprising:

- estimating a cost of merging a first set of instructions and a second set using

   a dataflow analysis (e.g., Abstract, 2nd Para., Lines 3-6 – based on a cost

   model, our algorithm groups instructions into thread by considering the trade-

   off among parallelism, latency tolerance, thread switching cost and sequential

   execution efficiency; Sec. 1.1 – An Example of Thread Partitioning, 1st Para.,

   Lines 1-9, 13-14; 1.2 – Synopsis, 2nd Para., Lines 6-19; Sec. 2.2 – Program

   Modeling; Sec. 6 of Related Work, 2nd Para., Lines 1-3 – thread partitioning

   for multi-threaded execution models has been mainly done for functional

   languages within the data-flow community; Reference, [4] – Integrating global

   caches and dataflow architecture, [5] – An evaluation of coarse-gain dataflow

code generation strategies, and [13] – A dataflow/von Neumann hybrid

architecture); and

- merging the first and second sets of instructions to form a merged set based

    on the cost of merging the first and second sets of instructions (e.g., Sec. 3 –

    Problem Statement, 8$^{th}$ Para., Lines 7-10 – two nodes can be merged into

    one thread to save thread switching costs; Sec. 4.1 – Overview of Thread

    Partitioning Heuristics, 4$^{th}$ Para., Lines 4-10 – We need to introduce the

    starting and finishing times of threads,....are merged into one thread; Sec.

    4.2.1 – Thread Formation, 1$^{st}$ Para., Lines 2-6 – merging nodes reduces the

    number of threads generated and thus reduce the total coast due to thread

    switching; Sec. 5.5 – Thread Length, 4$^{th}$ Para., 4-6 - ...which merges a node

    into the same thread with one of its local predecessors, applies nearly 50% of

    time overall).

Tang does not explicitly disclose wherein the cost of merging the first and

second sets of instructions is associated with instructions that belong to only the

first set of instructions and instructions that belong to only the second set of

instructions.

However, in an analogous art of *Sec. 4.1 Managing Lock Contention, Large*

*and Small Critical Sections, Chap. 4, Synchronization, Developing Multithreaded*

*Applications: A Platform Consistent Approach*, Haab discloses wherein the cost

of merging the first and second sets of instructions is associated with instructions

that belong to only the first set of instructions and instructions that belong to only

the second set of instructions (e.g., P. 75, Example Code 5 – A threaded

containing two critical sections (e.g., the 1st set of instructions –

CRITICAL_SECTION_1 and the 2nd set of instructions –

CRITICAL_SECTION_2) to protect updates to different shared data; Example

Code 6 – A threaded function containing one critical section (e.g., merging critical

sections) that protects updates to all shared data used by the function; P. 74,

Sec. Background, 2nd Par. - If the threads only spend a small amount of time in

DoFunc1, the synchronization overhead of two critical sections may not be

justified (e.g., based on cost analysis for each set of instructions). In this case, a

better scheme (e.g., a result of cost analysis – merging two set of instructions)

might be to merge the two small critical sections into one larger critical section,

as in Example Code 6; P. 75, 1st Par. – On execution, the thread that gains

access to the critical section spends a considerable amount of time in the critical

section (e.g., an associated cost with the 1st set of instructions), while all the

remaining threads are blocked (e.g., an associated cost with the 2nd set of

instructions). When the thread holding the lock releases it, one of the waiting

threads is allowed to enter the critical section and all other waiting threads

remain blocked for a long time. Therefore, two critical sections, as in Example

Code 5, is a better solution for this case (e.g., a result of cost analysis – not

merging two set of instructions); P. 77, Sec. Advice, 1st Par. – Balance the size of

critical sections against the overhead of acquiring and releasing locks).

Therefore, it would have been obvious to one of ordinary skill in the art, at

the time the invention was made to combine the teachings of Haab into the

Tang's system to further provide wherein the cost of merging the first and second

sets of instructions is associated with instructions that belong to only the first set of instructions and instructions that belong to only the second set of instructions in Tang system.

The motivation is that it would further enhance the Tang's system by taking, advancing and/or incorporating Haab's system which offers significant advantages that balance the size of critical sections against the overhead of acquiring and releasing locks; And, the optimum probably lies somewhere between the extremes of a different lock for every shared datum and a single lock for all shared data as once suggested by Haab (e.g., P. 77, Sec. Advice, 1$^{st}$ Par.).

10. **As to claim 2** (Original) (incorporating the rejection in claim 1), Tang discloses a method further comprising: estimating a cost of merging the first set of instructions and a third set of instructions; and estimating a cost of merging the second set of instructions and the third set of instructions (e.g., Sec. 2.2 – Program Modeling).

11. **As to claim 3** (Original) (incorporating the rejection in claim 2), Tang discloses a method wherein the cost of merging the first and third sets of instructions and the cost of merging the second and third sets of instructions are greater than the cost of merging the first and second sets of instructions (Sec. 2.2 – Program Modeling).

12.    **As to claim 4** (Currently Amended) (incorporating the rejection in claim 2),

Tang discloses a method wherein the third set of instructions comprises a critical

section of instructions (e.g., Fig. 4 – An Annotated DDG and Its Optimum

Partition; Sec. 3 – Problem Statement, 7th Para., Lines 1-4 – the difficulty in such

scheduling is that the critical path of the partitioned threads is unknown ..., 6-12;

Sec. 4.1 – Overview of Thread Partitioning Heuristics, 1st Para., 1-7 – To help

determine which nodes are critical,...., 3rd Para., Lines 3-10 – However, since $h_4$

is greater than both $h_2$ and $h_3$, we may ....on the critical path of final generated

threads; Sec. 4.2.2 – Sequencing within Threads, 2nd Para., Lines 7-12 –

However, $v_j$ itself could be on the critical path,...).

13.    **As to claim 5** (Currently Amended) (incorporating the rejection in claim 2),

Tang discloses a method wherein the third set of instructions is associated with a

critical section (e.g., Fig. 4 – An Annotated DDG and Its Optimum Partition; Sec.

3 – Problem Statement, 7th Para., Lines 1-4 – the difficulty in such scheduling is

that the critical path of the partitioned threads is unknown ..., 6-12; Sec. 4.1 –

Overview of Thread Partitioning Heuristics, 1st Para., 1-7 – To help determine

which nodes are critical,...., 3rd Para., Lines 3-10 – However, since $h_4$ is greater

than both $h_2$ and $h_3$, we may ....on the critical path of final generated threads;

Sec. 4.2.2 – Sequencing within Threads, 2nd Para., Lines 7-12 – However, $v_j$

itself could be on the critical path,...).

14.    **As to claim 8** (Original) (incorporating the rejection in claim 1), Tang

discloses a method wherein the first and second sets of instructions are

associated with respective first and second critical sections (e.g., Fig. 4 – An

Annotated DDG and Its Optimum Partition; Sec. 3 – Problem Statement, 7th

Para., Lines 1-4 – the difficulty in such scheduling is that the critical path of the

partitioned threads is unknown ..., 6-12; Sec. 4.1 – Overview of Thread

Partitioning Heuristics, 1st Para., 1-7 – To help determine which nodes are

critical,...., 3rd Para., Lines 3-10 – However, since $h_4$ is greater than both $h_2$ and

$h_3$, we may ....on the critical path of final generated threads; Sec. 4.2.2 –

Sequencing within Threads, 2nd Para., Lines 7-12 – However, $v_j$ itself could be on

the critical path,...).

15.    **As to claim 9** (Currently Amended) (incorporating the rejection in claim 1),

Tang discloses a method wherein at least one of the first and second sets of

instructions is associated with a critical section (e.g., Fig. 4 – An Annotated DDG

and Its Optimum Partition; Sec. 3 – Problem Statement, 7th Para., Lines 1-4 – the

difficulty in such scheduling is that the critical path of the partitioned threads is

unknown ..., 6-12; Sec. 4.1 – Overview of Thread Partitioning Heuristics, 1st

Para., 1-7 – To help determine which nodes are critical,...., 3rd Para., Lines 3-10

– However, since $h_4$ is greater than both $h_2$ and $h_3$, we may ....on the critical path

of final generated threads; Sec. 4.2.2 – Sequencing within Threads, 2nd Para.,

Lines 7-12 – However, $v_j$ itself could be on the critical path,...).

16.    **As to claim 16** (Original) (incorporating the rejection in claim 1), Tang

discloses a method further comprising creating a partition including the first and

the second sets of instructions before the first and second sets of instructions are

merged (e.g., Sec. 1.1 – An Example of Thread Partitioning, 1st Para., Lines 1-9,

13-14; 1.2 – Synopsis, 2nd Para., Lines 6-19; Sec. 2.2 – Program Modeling).

17.    **As to claim 17** (Currently Amended), Tang discloses an apparatus

comprising:

- an instruction analysis module configured to perform a dataflow analysis (e.g.,

  Sec. 4 – List-Scheduling Based Heuristic Algorithm; Sec. 6 – Related Work,

  2nd Para.; Abstract, 2nd Para., Lines 3-6 – based on a cost model, our

  algorithm groups instructions into thread by considering the trade-off among

  parallelism, latency tolerance, thread switching cost and sequential execution

  efficiency; Sec. 1.1 – An Example of Thread Partitioning, 1st Para., Lines 1-9,

  13-14; 1.2 – Synopsis, 2nd Para., Lines 6-19; Sec. 6 of Related Work, 2nd

  Para., Lines 1-3 – thread partitioning for multi-threaded execution models has

  been mainly done for functional languages within the data-flow community;

  Reference, [4] – Integrating global caches and dataflow architecture, [5] – An

  evaluation of coarse-gain dataflow code generation strategies, and [13] – A

  dataflow/von Neumann hybrid architecture);

- a cost estimation module configured to determine an estimated cost of

  merging a first set of instructions and a second set of instructions to form a

  merged set of instructions (e.g., Sec. 2.2 – Program Modeling); and

- a partition generator (e.g., Sec. 1.1 – An Example of Thread Partitioning) configured to merge the first and second sets of instructions based on the estimated cost of merging the first and second sets of instructions (e.g., Sec. 3 – Problem Statement, 8th Para., Lines 7-10 – two nodes can be merged into one thread to save thread switching costs; Sec. 4.1 – Overview of Thread Partitioning Heuristics, 4th Para., Lines 4-10 – We need to introduce the starting and finishing times of threads,....are merged into one thread; Sec. 4.2.1 – Thread Formation, 1st Para., Lines 2-6 – merging nodes reduces the number of threads generated and thus reduce the total coast due to thread switching; Sec. 5.5 – Thread Length, 4th Para., 4-6 - ...which merges a node into the same thread with one of its local predecessors, applies nearly 50% of time overall).

Tang does not explicitly disclose wherein the cost of merging the first and second sets of instructions is associated with instructions that belong to only the first set of instructions and instructions that belong to only the second set of instructions.

However, in an analogous art of *Sec. 4.1 Managing Lock Contention, Large and Small Critical Sections, Chap. 4, Synchronization, Developing Multithreaded Applications: A Platform Consistent Approach*, Haab discloses wherein the cost of merging the first and second sets of instructions is associated with instructions that belong to only the first set of instructions and instructions that belong to only the second set of instructions (e.g., P. 75, Example Code 5 – A threaded containing two critical sections (e.g., the 1st set of instructions –

CRITICAL_SECTION_1 and the 2<sup>nd</sup> set of instructions –

CRITICAL_SECTION_2) to protect updates to different shared data; Example

Code 6 – A threaded function containing one critical section (e.g., merging critical

sections) that protects updates to all shared data used by the function; P. 74,

Sec. Background, 2<sup>nd</sup> Par. - If the threads only spend a small amount of time in

DoFunc1, the synchronization overhead of two critical sections may not be

justified (e.g., based on cost analysis for each set of instructions). In this case, a

better scheme (e.g., a result of cost analysis – merging two set of instructions)

might be to merge the two small critical sections into one larger critical section,

as in Example Code 6; P. 75, 1<sup>st</sup> Par. – On execution, the thread that gains

access to the critical section spends a considerable amount of time in the critical

section (e.g., an associated cost with the 1<sup>st</sup> set of instructions), while all the

remaining threads are blocked (e.g., an associated cost with the 2<sup>nd</sup> set of

instructions). When the thread holding the lock releases it, one of the waiting

threads is allowed to enter the critical section and all other waiting threads

remain blocked for a long time. Therefore, two critical sections, as in Example

Code 5, is a better solution for this case (e.g., a result of cost analysis – not

merging two set of instructions); P. 77, Sec. Advice, 1<sup>st</sup> Par. – Balance the size of

critical sections against the overhead of acquiring and releasing locks).

Therefore, it would have been obvious to one of ordinary skill in the art, at

the time the invention was made to combine the teachings of Haab into the

Tang's system to further provide wherein the cost of merging the first and second

sets of instructions is associated with instructions that belong to only the first set

of instructions and instructions that belong to only the second set of instructions in Tang system.

The motivation is that it would further enhance the Tang's system by taking, advancing and/or incorporating Haab's system which offers significant advantages that balance the size of critical sections against the overhead of acquiring and releasing locks; And, the optimum probably lies somewhere between the extremes of a different lock for every shared datum and a single lock for all shared data as once suggested by Haab (e.g., P. 77, Sec. Advice, 1st Par.).

18.  **As to claim 22** (Original) (incorporating the rejection in claim 17), Tang discloses an apparatus wherein the partition generator is configured to create a partition including the first and second sets of instructions before the first and second sets of instructions are merged (e.g., Sec. 1.1 – An Example of Thread Partitioning, 1st Para., Lines 1-9, 13-14; 1.2 – Synopsis, 2nd Para., Lines 6-19; Sec. 2.2 – Program Modeling).

19.  **As to claim 25** (Currently Amended), Tang discloses a machine readable medium having instructions stored thereon that, when executed, cause a machine to:

- estimate a cost of merging a first set of instructions and a second set of instructions using a dataflow analysis (e.g., Abstract, 2nd Para., Lines 3-6 – based on a cost model, our algorithm groups instructions into thread by

considering the trade-off among parallelism, latency tolerance, thread

switching cost and sequential execution efficiency; Sec. 1.1 – An Example of

Thread Partitioning, 1st Para., Lines 1-9, 13-14; 1.2 – Synopsis, 2nd Para.,

Lines 6-19; Sec. 2.2 – Program Modeling; Sec. 6 of Related Work, 2nd Para.,

Lines 1-3 – thread partitioning for multi-threaded execution models has been

mainly done for functional languages within the data-flow community;

Reference, [4] – Integrating global caches and dataflow architecture, [5] – An

evaluation of coarse-gain dataflow code generation strategies, and [13] – A

dataflow/von Neumann hybrid architecture); and

- merge the first and the second sets of instructions to form a merged set of

  instructions based on the cost of merging the first and second sets of

  instructions (e.g., Sec. 3 – Problem Statement, 8th Para., Lines 7-10 – two

  nodes can be merged into one thread to save thread switching costs; Sec. 4.1

  – Overview of Thread Partitioning Heuristics, 4th Para., Lines 4-10 – We need

  to introduce the starting and finishing times of threads,....are merged into one

  thread; Sec. 4.2.1 – Thread Formation, 1st Para., Lines 2-6 – merging nodes

  reduces the number of threads generated and thus reduce the total coast due

  to thread switching; Sec. 5.5 – Thread Length, 4th Para., 4-6 - ...which

  merges a node into the same thread with one of its local predecessors,

  applies nearly 50% of time overall).

Tang does not explicitly disclose wherein the cost of merging the first and

second sets of instructions is associated with instructions that belong to only the

first set of instructions and instructions that belong to only the second set of

instructions.

However, in an analogous art of *Sec. 4.1 Managing Lock Contention, Large

and Small Critical Sections, Chap. 4, Synchronization, Developing Multithreaded

Applications: A Platform Consistent Approach*, Haab discloses wherein the cost

of merging the first and second sets of instructions is associated with instructions

that belong to only the first set of instructions and instructions that belong to only

the second set of instructions (e.g., P. 75, Example Code 5 – A threaded

containing two critical sections (e.g., the 1st set of instructions –

CRITICAL_SECTION_1 and the 2nd set of instructions –

CRITICAL_SECTION_2) to protect updates to different shared data; Example

Code 6 – A threaded function containing one critical section (e.g., merging critical

sections) that protects updates to all shared data used by the function; P. 74,

Sec. Background, 2nd Par. - If the threads only spend a small amount of time in

DoFunc1, the synchronization overhead of two critical sections may not be

justified (e.g., based on cost analysis for each set of instructions). In this case, a

better scheme (e.g., a result of cost analysis – merging two set of instructions)

might be to merge the two small critical sections into one larger critical section,

as in Example Code 6; P. 75, 1st Par. – On execution, the thread that gains

access to the critical section spends a considerable amount of time in the critical

section (e.g., an associated cost with the 1st set of instructions), while all the

remaining threads are blocked (e.g., an associated cost with the 2nd set of

instructions). When the thread holding the lock releases it, one of the waiting

threads is allowed to enter the critical section and all other waiting threads remain blocked for a long time. Therefore, two critical sections, as in Example Code 5, is a better solution for this case (e.g., a result of cost analysis – not merging two set of instructions); P. 77, Sec. Advice, 1st Par. – Balance the size of critical sections against the overhead of acquiring and releasing locks).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Haab into the Tang's system to further provide wherein the cost of merging the first and second sets of instructions is associated with instructions that belong to only the first set of instructions and instructions that belong to only the second set of instructions in Tang system.

The motivation is that it would further enhance the Tang's system by taking, advancing and/or incorporating Haab's system which offers significant advantages that balance the size of critical sections against the overhead of acquiring and releasing locks; And, the optimum probably lies somewhere between the extremes of a different lock for every shared datum and a single lock for all shared data as once suggested by Haab (e.g., P. 77, Sec. Advice, 1st Par.).

20.   **As to claim 26** (Original) (incorporating the rejection in claim 25), Tang discloses a machine readable medium having instructions stored thereon that, when executed, cause the machine to: estimate a cost of merging the first set of instructions and a third set of instructions; and estimate a cost of merging the

second set of instructions and the third set of instructions (e.g., Sec. 2.2 –

Program Modeling).

21. **As to claim 30** (Original) (incorporating the rejection in claim 25), Tang

discloses a machine readable medium having instructions stored thereon that,

when executed, cause the machine to create a partition including the first and

second sets of instructions before the first and second sets of instructions are

merged (e.g., Sec. 1.1 – An Example of Thread Partitioning, 1$^{st}$ Para., Lines 1-9,

13-14; 1.2 – Synopsis, 2$^{nd}$ Para., Lines 6-19; Sec. 2.2 – Program Modeling).

22. Claims 6, 18, and 27 are rejected under 35 U.S.C. 103(a) as being

unpatentable over Tang in view of Haab and further in view of Zoppetti et al.,

(*Automatic Compiler Techniques for Thread Coarsening for Multithreaded*

*Architectures*) (hereinafter 'Zoppetti') and Buch et al., (Pub. No. US

2003/0065704 A1) (hereinafter 'Buch')

23. **As to claim 6** (Original) (incorporating the rejection in claim 1), Tang

discloses thread partitioning based on cost model (e.g., Abstract, 2$^{nd}$ Para.), but

Tang and Haab do not explicitly disclose a method further comprising: removing

redundant instructions from the merged set of instructions; and assigning a

physical mutual exclusion lock to the merged set of instructions.

However, in an analogous art of *automatic compiler techniques for thread*

*coarsening for multithreaded architectures*, Zoppetti discloses a method further

comprising: removing redundant instructions from the merged set of instructions

(Sec. 1 – Introduction, 6th Para., 3rd bullet – use of must alias or definite points-to

information for removing redundant remote references; Sec. 3.3 – Using Must

Alias Information, 3rd Para., Lines 1-3; Sec. 6, 1st Para., Lines 6-9).

Therefore, it would have been obvious to one of ordinary skill in the art, at

the time the invention was made to combine the teachings of Zoppetti into the

Tang-Haab's system to provide a method further comprising: removing

redundant instructions from the merged set of instructions in Tang-Haab's

system.

The motivation is that it would further enhance the Tang-Haab's system by

advantageously taking, advancing and/or incorporating Zoppetti's system which

provides the techniques that lead to improve extraction and representation of

dependence information in the presence of structured control flow....; The benefit

of these techniques is the generation of coarser-gained threads and, therefore,

decreased execution tome as once suggested by Zoppetti (e.g., Abstract; 2nd

Para.).

Further, Tang discloses critical section of the partitioned threads (Sec. 3 –

Problem Statement, 7th Para.), but Tang, Haab and Zoppetti do not explicitly

disclose assigning a physical mutual exclusion lock to the merged set of

instructions.

However, in an art of *flexible acceleration of Java™ thread*

*synchronization on multiprocessor computers*, Buch discloses assigning a

physical mutual exclusion lock to the merged set of instructions (e.g., [0022],

Lines 5-8 - .. a hardware lock has been assigned...).

Therefore, it would have been obvious to one of ordinary skill in the art, at

the time the invention was made to combine the teachings of Buch into the Tang-

Haab-Zoppetti's system to further assign a physical mutual exclusion lock to the

merged set of instructions in Tang-Haab-Zoppetti's system.

The motivation is that it would enhance the Tang-Haab-Zoppetti's system

by taking, advancing and/or incorporating Buch's system which discloses that

shared resources have associated "locks." A thread must acquire the lock on a

resource in order to access the resource as once suggested by Buch (e.g.,

[0002], Lines 6-11).

24.     **As to claim 18** (Original) (incorporating the rejection in claim 17), Tang

discloses thread partitioning based on cost model (e.g., Abstract, 2nd Para.), but

Tang and Haab do not explicitly disclose an apparatus as defined further

comprising: a redundant instruction module configured to remove redundant

instructions from the merged set of instructions; and a mutual exclusion lock

module configured to assign a first physical mutual exclusion lock to the merged

set of instructions.

However, in an analogous art of *automatic compiler techniques for thread*

*coarsening for multithreaded architectures*, Zoppetti discloses an apparatus as

defined further comprising: a redundant instruction module configured to remove

redundant instructions from the merged set of instructions (e.g., Sec. 1 –

Introduction, 6<sup>th</sup> Para., 3<sup>rd</sup> bullet – use of must alias or definite points-to

information for removing redundant remote references; Sec. 3.3 – Using Must

Alias Information, 3<sup>rd</sup> Para., Lines 1-3; Sec. 6, 1<sup>st</sup> Para., Lines 6-9).

Therefore, it would have been obvious to one of ordinary skill in the art, at

the time the invention was made to combine the teachings of Zoppetti into the

Tang-Haab's system to provide an apparatus as defined further comprising: a

redundant instruction module configured to remove redundant instructions from

the merged set of instructions in Tang-Haab's system.

The motivation is that it would further enhance the Tang-Haab's system by

advantageously taking, advancing and/or incorporating Zoppetti's system which

provides the techniques that lead to improve extraction and representation of

dependence information in the presence of structured control flow....; The benefit

of these techniques is the generation of coarser-gained threads and, therefore,

decreased execution tome as once suggested by Zoppetti (e.g., Abstract; 2<sup>nd</sup>

Para.).

Further, Tang discloses critical section of the partitioned threads (Sec. 3 –

Problem Statement, 7<sup>th</sup> Para.), but both Tang, Haab and Zoppetti do not explicitly

disclose a mutual exclusion lock module configured to assign a first physical

mutual exclusion lock to the merged set of instructions.

However, in an art of *flexible acceleration of Java™ thread*

*synchronization on multiprocessor computers*, Buch discloses a mutual exclusion

lock module configured to assign a first physical mutual exclusion lock to the

merged set of instructions (e.g., [0022], Lines 5-8 - .. a hardware lock has been assigned...).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Buch into the Tang-Haab-Zoppetti's system to further provide a mutual exclusion lock module configured to assign a first physical mutual exclusion lock to the merged set of instructions in Tang-Haab-Zoppetti's system.

The motivation is that it would enhance the Tang-Haab-Zoppetti's system by taking, advancing and/or incorporating Buch's system which discloses that shared resources have associated "locks." A thread must acquire the lock on a resource in order to access the resource as once suggested by Buch (e.g., [0002], Lines 6-11).

25.  **As to claim 27** (Original) (incorporating the rejection in claim 25), Tang discloses thread partitioning based on cost model (e.g., Abstract, 2nd Para.), but Tang and Haab do not explicitly disclose a machine readable medium having instructions stored thereon that, when executed, cause the machine to: remove redundant instructions from the merged set of instructions; and assign a physical mutual exclusion lock to the merged set of instructions.

However, in an analogous art of *automatic compiler techniques for thread coarsening for multithreaded architectures,* Zoppetti discloses a machine readable medium having instructions stored thereon that, when executed, cause the machine to: remove redundant instructions from the merged set of

instructions (e.g., Sec. 1 – Introduction, 6<sup>th</sup> Para., 3<sup>rd</sup> bullet – use of must alias or definite points-to information for removing redundant remote references; Sec. 3.3 – Using Must Alias Information, 3<sup>rd</sup> Para., Lines 1-3; Sec. 6, 1<sup>st</sup> Para., Lines 6-9).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Zoppetti into the Tang-Haab's system to provide a machine readable medium having instructions stored thereon that, when executed, cause the machine to: remove redundant instructions from the merged set of instructions in Tang-Haab's system.

The motivation is that it would further enhance the Tang-Haab's system by advantageously taking, advancing and/or incorporating Zoppetti's system which provides the techniques that lead to improve extraction and representation of dependence information in the presence of structured control flow....; The benefit of these techniques is the generation of coarser-gained threads and, therefore, decreased execution tome as once suggested by Zoppetti (e.g., Abstract; 2<sup>nd</sup> Para.).

Further, Tang discloses critical section of the partitioned threads (e.g., Sec. 3 – Problem Statement, 7<sup>th</sup> Para.), but Tang, Haab and Zoppetti do not explicitly disclose assigning a physical mutual exclusion lock to the merged set of instructions.

However, in an art of *flexible acceleration of Java™ thread synchronization on multiprocessor computers*, Buch discloses assigning a physical mutual exclusion lock to the merged set of instructions (e.g., [0022], Lines 5-8 - .. a hardware lock has been assigned...).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Buch into the Tang-Haab-Zoppetti's system to further assign a physical mutual exclusion lock to the merged set of instructions in Tang-Haab-Zoppetti's system.

The motivation is that it would enhance the Tang-Zoppetti's system by taking, advancing and/or incorporating Buch's system which discloses that shared resources have associated "locks." A thread must acquire the lock on a resource in order to access the resource as once suggested by Buch (e.g., [0002], Lines 6-11).

26.    Claims 7 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tang in view of Haab, Zoppetti and Buch and further in view of T. Ogasawara et al., (Pat. No. US 7,089,540 B2) (hereinafter 'Ogasawara')

27.    **As to claim 7** (Original) (incorporating the rejection in claim 6), Tang discloses thread partitioning based on cost model (e.g., Abstract, 2nd Para.) and Zoppetti discloses removing redundant instructions (e.g., Sec. 1 – Introduction, 6th Para., 3rd bullet – use of must alias or definite points-to information for removing redundant remote references; Sec. 3.3 – Using Must Alias Information) but Tang, Haab, Zoppetti and Buch do not explicitly disclose a method wherein the redundant instructions comprise instructions used for at least one of entering a set of instructions and exiting the set of instructions.

However, in an analogous art of *automatic compiler techniques for thread coarsening for multithreaded architectures*, Ogasawara discloses a method wherein the redundant instructions comprise instructions used for at least one of entering a set of instructions and exiting the set of instructions (e.g., Fig. 2, element 140 – Synchronization Optimization Unit; Col. 3, Lines 30-32, 64-67; Col. 4, 36-39; Col. 9, Lines 16-24; Col. 12, Lines 48-50).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Ogasawara into the Tang-Haab-Zoppetti-Buch's system to provide a method wherein the redundant instructions comprise instructions used for at least one of entering a set of instructions and exiting the set of instructions in Tang-Haab-Zoppetti-Buch's system.

The motivation is that it would further enhance the Tang-Haab-Zoppetti-Buch's system by advantageously taking, advancing and/or incorporating Ogasawara's system which provides a synchronization optimization unit for deleting an unnecessary lock from the target program as once suggested by Ogasawara (e.g., Fig. 2, element 140 – Synchronization Optimization Unit; Col. 3, Lines 30-32, 64-67; Col. 4, 36-39).

28.     **As to claim 20** (Original) (incorporating the rejection in claim 18), Tang discloses thread partitioning based on cost model (e.g., Abstract, 2nd Para.) and Zoppetti discloses removing redundant instructions (e.g., Sec. 1 – Introduction, 6th Para., 3rd bullet – use of must alias or definite points-to information for

removing redundant remote references; Sec. 3.3 – Using Must Alias Information)

but Tang, Haab, Zoppetti and Buch do not explicitly disclose an apparatus

wherein the redundant instruction module is configured to remove redundant

instructions comprising instructions for at least one of entering a set of

instructions and exiting the set of instructions.

However, in an analogous art of *automatic compiler techniques for thread*

*coarsening for multithreaded architectures*, Ogasawara discloses an apparatus

wherein the redundant instruction module is configured to remove redundant

instructions comprising instructions for at least one of entering a set of

instructions and exiting the set of instructions (e.g., Fig. 2, element 140 –

Synchronization Optimization Unit; Col. 3, Lines 30-32, 64-67; Col. 4, 36-39; Col.

9, Lines 16-24; Col. 12, Lines 48-50).

Therefore, it would have been obvious to one of ordinary skill in the art, at

the time the invention was made to combine the teachings of Ogasawara into the

Tang-Haab-Zoppetti-Buch's system to provide an apparatus wherein the

redundant instruction module is configured to remove redundant instructions

comprising instructions for at least one of entering a set of instructions and

exiting the set of instructions in Tang-Haab-Zoppetti-Buch's system.

The motivation is that it would further enhance the Tang-Haab-Zoppetti-

Buch's system by advantageously taking, advancing and/or incorporating

Ogasawara's system which provides a synchronization optimization unit for

deleting an unnecessary lock from the target program as once suggested by

Ogasawara (e.g., Fig. 2, element 140 – Synchronization Optimization Unit; Col.

3, Lines 30-32, 64-67; Col. 4, 36-39).


29.    Claims 10 and 21 are rejected under 35 U.S.C. 103(a) as being

unpatentable over Tang in view of Haab and further in view of Moon et al.,

(*Evaluation of Predicated Array Data-Flow Analysis for Automatic Parallelization,*

*1999 ACM*) (hereinafter 'Moon')


30.    **As to claim 10** (Original) (incorporating the rejection in claim 1), Tang

discloses using a dataflow analysis (e.g., Sec. 6 of Related Work, 2nd Para.) but

Tang and Haab do not explicitly disclose a method wherein the dataflow analysis

comprises a forward disjunctive dataflow analysis.

However, in an analogous art of *automatic compiler techniques for thread*

*coarsening for multithreaded architectures*, Moon discloses a method wherein

the dataflow analysis comprises a forward disjunctive dataflow analysis (e.g.,

Sec. 4.1 – Predicate Domain, 4th Para., Lines 1-4, 5th Para., Lines 1-5).

Therefore, it would have been obvious to one of ordinary skill in the art, at

the time the invention was made to combine the teachings of Moon into the

Tang-Haab's system to provide a method wherein the dataflow analysis

comprises a forward disjunctive dataflow analysis in Tang-Haab's system.

The motivation is that it would enhance the Tang-Haab's system by taking,

advancing and/or incorporating Moon's system which provides two distinguished

features (1) it derives low-cost, run-time parallelization test; and, (2) it

incorporates predicate embedding and predicate extraction, which translate

between the domain of predicates and data-flow values to derive more precise

analysis results as once suggested by Moon (e.g., Abstract; 2nd Para., Lines 2-7).

31.    **As to claim 21** (Original) (incorporating the rejection in claim 17), Tang

discloses using a dataflow analysis (e.g., Sec. 6 of Related Work, 2nd Para.) but

Tang and Haab do not explicitly disclose an apparatus wherein the instruction

analysis module is configured to perform a forward disjunctive dataflow analysis.

However, in an analogous art of *automatic compiler techniques for thread*

*coarsening for multithreaded architectures*, Moon discloses an apparatus

wherein the instruction analysis module is configured to perform a forward

disjunctive dataflow analysis (e.g., Sec. 4.1 – Predicate Domain, 4th Para., Lines

1-4, 5th Para., Lines 1-5).

Therefore, it would have been obvious to one of ordinary skill in the art, at

the time the invention was made to combine the teachings of Moon into the

Tang-Haab's system to provide an apparatus wherein the instruction analysis

module is configured to perform a forward disjunctive dataflow analysis in Tang-

Haab's system.

The motivation is that it would enhance the Tang-Haab's system by taking,

advancing and/or incorporating Moon's system which provides two distinguished

features (1) it derives low-cost, run-time parallelization test; and, (2) it

incorporates predicate embedding and predicate extraction, which translate

between the domain of predicates and data-flow values to derive more precise analysis results as once suggested by Moon (e.g., Abstract; 2nd Para., Lines 2-7).

## *Allowable Subject Matter*

32.    Claims 12-15, 19, 23-24, and 28-29 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten to overcome all the limitations of the base claim and any intervening claims.

The following is an examiner's statement of reasons for allowance:

33.    **Regarding claims** 12-15, 19, 23-24, and 28-29, prior art of record fails to reasonably show or suggest the specific vectors created based on the dataflow analysis, wherein elements of the first vector comprise instructions contained in at least one of the first and second sets of instructions and the cost matrix created based on the first vector, wherein the cost matrix contains the cost of merging the first and second sets of instructions as claimed. Further, the second vector comprising a redundancy indicator after merging the first and the second set of instructions, furthermore updating the first vector and the cost matrix after merging the first and second sets of instructions.

## *Conclusion*

34.    Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is

571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00

a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the

examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax

phone number for the organization where this application or proceeding is

assigned is 571-273-8300.

Information regarding the status of an application may be obtained from

the Patent Application Information Retrieval (PAIR) system. Status information

for published applications may be obtained from either Private PAIR or Public

PAIR. Status information for unpublished applications is available through

Private PAIR only. For more information about the PAIR system, see http://pair-

direct.uspto.gov. Should you have questions on access to the Private PAIR

system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-

free). If you would like assistance from a USPTO Customer Service

Representative or access to the automated information system, call 800-786-

9199 (IN USA OR CANADA) or 571-272-1000.

BCW *BW*

TUAN DAM
SUPERVISORY PATENT EXAMINER

December 18, 2007